

**REMARKS/ARGUMENTS**

Claims 1, 2, 4-13, and 15-22 are rejected under 35 U.S.C. 103(a) as being unpatentable of Fischer (US Patent No. 6,823,353) in view of Kabir (US Patent No. 5,933,160). Claims 3 and 14 are rejected under 35 U.S.C. 103(a) as being unpatentable over Fischer and Kabir, and further in view of Cohen (US Patent No. 5,751,614). Claims 33, 35, 40-42, 45, and 50-52 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kabir. Claims 34, 36-39, 44, and 46-49 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kabir and further in view of Thekkath (US Patent No. 6,732,259). Claims 1, 2, 7, 9, 11-13, 18, 20, and 22 have been amended to correct minor typographical errors and improve readability.

Applicants respectfully traverse the pending § 103 rejections based on various combinations of Fischer, Kabir, Cohen, and Thekkath, because none of these cited references qualifies as prior art against the present invention. The present application claims priority back to the 8/16/95 filing date of U.S. Patent Application No. 08/516,036, which issues as U.S. Patent No. 5,742,840 (the '840 patent). This chain of priority also includes a continuation-in-part (CIP) application, U.S. Patent Application No. 09/382,402, which issued as U.S. Patent No. 5,295,599 (the '599 patent).

Fischer, Kabir, Cohen, and Thekkath are all dated after the 8/16/95 priority date of the present invention. Fischer is a U.S. patent that has an earliest possible effective filing date of 9/5/95 and an issue date of 11/23/04.<sup>1</sup> Kabir is a U.S. patent that has a filing date of 11/27/95 and an issue date of 8/3/99. Cohen is a U.S. patent that has an earliest possible effective filing date of 2/29/96 and an issue date of 5/12/98.<sup>2</sup> Thekkath is a U.S. patent that has an filing date of 7/30/99 and an issue date of 5/4/04. As such, none of the cited references qualifies as prior art

---

<sup>1</sup> Fischer is a continuation of application Ser. No. 09/760,969, filed 1/16/01, which claims priority to a divisional application Ser. No. 08/905,506, filed 7/31/97, which claims priority to application Ser. No. 08/575,778, filed 12/20/95, which is a continuation-in-part of and claims priority to Ser. No. 08/523,211, filed 9/5/95.

<sup>2</sup> Cohen fails to qualify as prior art because the earliest filing date of Cohen associated with the feature cited by the Examiner is 2/29/96. The Examiner cites to the "mask" feature in Fig. 3 of Cohen. Cohen is a continuation-in-part (CIP) of parent application 08/444,814, filed 5/18/95. However, the "mask" feature cited by the Examiner was not disclosed in the parent application. That is, the "mask" feature appeared for the first time in the Cohen application

against the presently claimed invention, and the § 103 rejections based on these references should be withdrawn.

The present invention is entitled to the 8/16/95 filing date for at least two important reasons. First, the '840 and '599 disclosures provide support for the claimed invention, contrary to the Examiner's assertions. Second, the support for the claimed invention found in the '840 and '599 disclosures clearly satisfies the requirement for written description and enablement under 35 U.S.C. § 112, paragraph 1. These reasons are explained in detail below.

#### **I. THE '840 AND '599 DISCLOSURES PROVIDE SUPPORT FOR THE CLAIMED INVENTION**

Contrary to the Examiner's assertions, the '840 and '599 disclosures provide support for the claimed invention. The Examiner questions the 8/16/95 filing date of the present invention, by alleging that no support for the claimed invention can be found in the '840 and '599 disclosures. The Examiner contends that no "single instruction" is taught to perform a multiply-add operation. Specifically, the Examiner states:

**However, the multiply-add operation is not taught and no means for a single instruction to perform such a complex operation is taught by the '599 or '840 patents. Office Action dated 2/21/07, p. 10, last paragraph to p. 11, first paragraph.**

The Examiner further contends that no support is found for storing a plurality of floating-point values where each floating-point value is understood as an IEEE standard floating-point separate from the other floating-point values and defined portions such as a mantissa. Specifically, the Examiner states:

---

filed 2/29/96, which is after the 8/16/95 priority date of the present application. As such, Cohen fails to qualify as prior art against the pending claims. Kabir also fails to qualify as prior art.

**Therefore there is no support for the storing of plurality of floating point values where a floating point value is understood as an IEEE standard floating point value that would have been accessible separately from another floating point value. Without any teachings for plural floating point of other separately accessible data or indication of where the portions of the standard floating point number such as the mantissa reside in the register or memory location then the data is merely a grouping of bits that operated on in some fashion. *Id.*, p. 11, first paragraph.**

Applicants strongly disagree with the Examiner's contentions. The '840 and '599 disclosures clearly teach a single instruction that performs a "group" multiply-add operation using true, IEEE standard-conforming floating-point values. The '840 and '599 disclosures each includes an appendix filed as part of the original application. Both the '840 appendix and the '599 appendix describe embodiments of a "single instruction." that performs a group multiply-add operation.

In fact, at least three different embodiments of such a "single" instruction are described: (1) "GF.MULADD.16" (Group Floating-Point Multiply and Add Half Precision); (2) "GF.MULADD.32" (Group Floating-Point Multiply and Add Single Precision); and (3) "GF.MULADD.64" (Group Floating-Point Multiply and Add Double Precision). These three instructions are identified at pp. 136-137 of the '840 appendix, and at pp. 264-266 of the '599 appendix.<sup>3</sup> A listing of these three different embodiments from p. 136 of the '840 appendix is reproduced below (instructions highlighted):

Operation codes

GF.MULADD.16	Group floating-point multiply and add half
GF.MULSUB.16	Group floating-point multiply and subtract half
GF.MULADD.32	Group floating-point multiply and add single
GF.MULSUB.32	Group floating-point multiply and subtract single
GF.MULADD.64	Group floating-point multiply and add double
GF.MULSUB.64	Group floating-point multiply and subtract double

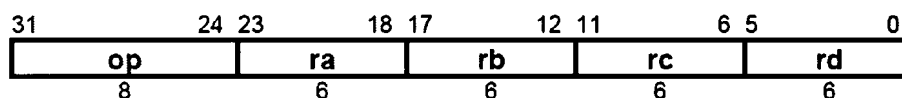
	op	prec		
multiply and add	MULADD	16	32	64
multiply and subtract	MULSUB	16	32	64

<sup>3</sup> In the '599 appendix, these three instructions are referred to as "Ensemble" instructions. They are: (1) "E.MUL.ADD.F.16" (Ensemble Multiply Add Floating-Point Half Precision); (2) "E.MUL.ADD.F.32" (Ensemble

As a specific example, the "GF.MULADD.32" (Group Floating-Point Multiply and Add Single Precision) instruction is discussed below. The "GF.MULADD.32" instruction is clearly a "single" instruction that performs a group multiply-add operation. The operation of the instruction is precisely defined in full detail down to the bit level, in a format that one of ordinary skill in the art would understand. Even though a group multiply-add operation using floating-point values is characterized by the Examiner as a "complex" operation, that is exactly what the "GF.MULADD.32" instruction does.<sup>4</sup> Specific portions of the disclosure that define this instruction as found in the '840 appendix at pp. 136-137 are reproduced below (highlighting added):<sup>5</sup>

#### Format

GF.operation.type      rd=ra,rb,rc



#### Description

The contents of registers or register pairs specified by ra and rb are multiplied together and added to or subtracted from the contents of the register or register pair specified by rc. The result is rounded to the nearest representable floating-point value in a single floating-point operation. The result is placed in the register or register pair specified by rd. Floating-point exceptions are not raised, and are handled according to the default rules of IEEE 754. These instructions cannot select a directed rounding mode or trap on inexact.

---

Multiply Add Floating-Point Single Precision); and (3) "E.MUL.ADD.64" (Ensemble Multiply Add Floating-Point Double Precision).

<sup>4</sup> See Office Action dated 2/21/07, p. 10, last paragraph to p. 11, first paragraph ("However, the multiply-add operation is not taught and no means for a single instruction to perform this complex operation is taught by the '599 or '840 patents").

<sup>5</sup> Similar portions of the definition for the comparable "E.MUL.ADD.32" instruction are found in the '599 appendix at pp. 264-266.

Definition

```
def GroupFloatingPointTernary(op,prec,ra,rb,rc,rd) as
  a ← RegRead(ra, 128)
  b ← RegRead(rb, 128)
  c ← RegRead(rc, 128)
  for i ← 0 to 128-prec by prec
    ai ← F(prec,ai+prec-1..i)
    bi ← F(prec,bi+prec-1..i)
    ci ← F(prec,ci+prec-1..i)
    case op of
      GF.MULADD:
        di ← (ai * bi) + ci
      GF.MULSUB:
        di ← (ai * bi) - ci
    endcase
    di+prec-1..i ← PackF(prec, di)
  endfor
  RegWrite(re, 128, d)
enddef
```

As shown above, the "GF.MULADD.32" instruction operates on groups of 32-bit, single-precision floating-point operands stored in 128-bit wide registers. Specifically, the "precision" utilized by this instruction is "32 bits," used to represent a "single-precision" floating-point number as defined by the IEEE 754 standard for floating-point number representation. A first group of 32-bit, single-precision floating-point operands are stored in register "ra." A second group of 32-bit, single-precision floating-point operands are stored in register "rb." A third group of 32-bit, single-precision floating-point operands are stored in register "rc." The "GF.MULADD.32" instruction operates by multiplying the first group of floating-point operands stored in register "ra" to the second group of floating-point operands stored in register "rb," to produce a group of products. The group of products is added to the third group of floating-point operands stored in register "rc," to produce a group of sums. In this example, the group of sums is provided as a catenated result to register "rd." This is a true "group" multiply-add operation that uses single-precision floating-point values.

Indeed, the '840 and '599 disclosures make it abundantly clear that the floating-point operands discussed therein indeed conform to the IEEE 754 standard for floating-point representation. Specifically, the 32-bit Specifically, the '840 and '599 disclosures provide detailed descriptions of the exact structure of each type of floating-point operand – including the

bit locations of the (1) "sign" field, (2) "exponent" field, and (3) "significand" field (also known as the "mantissa"):

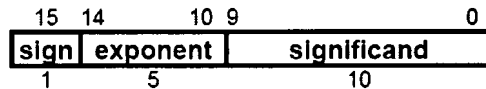
### Floating-point Data

Terpsichore's floating-point formats are designed to satisfy ANSI/IEEE standard 754-1985: Binary Floating-point Arithmetic. Standard 754 leaves certain aspects to the discretion of the implementor:

Terpsichore adds additional half-precision and quad-precision formats to standard 754's single-precision and double-precision formats. Terpsichore's double-precision satisfies standard 754's precision requirements for a single-extended format, and Terpsichore's quad-precision satisfies standard 754's precision requirements for a double-extended format.

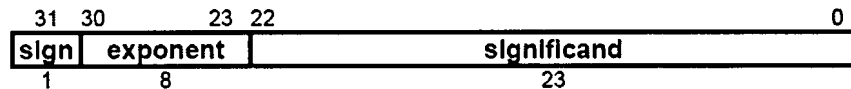
### Half-precision Floating-point

Terpsichore half precision uses a format similar to standard 754's requirements, reduced to a 16-bit overall format. The format contains sufficient precision and exponent range to hold a 12-bit signed integer.



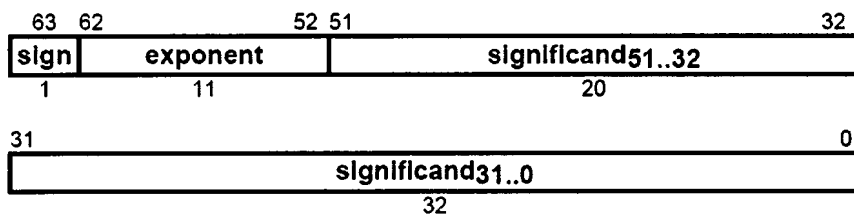
### Single-precision Floating-point

Terpsichore single precision satisfies standard 754's requirements for "single."



### Double-precision Floating-point

Terpsichore double precision satisfies standard 754's requirements for "double."



Accordingly, the '840 and '599 disclosures clearly describe a "single" instruction capable of performing a "group" multiply-add operation using multiple floating-point operands. The disclosures illustrate exactly how the groups floating-point operands can be accessed from specific registers to achieve the group multiply-add operation. The clear, bit-level definition of the format of each type of floating-point operand leaves no doubt as to how the individual

floating-point operands are formatted, in conformance to the IEEE 754 standard for floating-point representation. In light of the above, Applicants firmly believe that the claimed invention is properly supported by the '840 and '599 disclosures, contrary to the Examiner's contentions.

**II. SUPPORT FOR THE CLAIMED INVENTION FOUND IN THE '840 AND '599 DISCLOSURES CLEARLY SATISFIES THE REQUIREMENT FOR WRITTEN DESCRIPTION AND ENABLEMENT UNDER 35 U.S.C. § 112, PARAGRAPH 1**

Furthermore, support for the claimed invention found in the '840 and '599 disclosures satisfies the requirements for written description and enablement under 35 U.S.C. § 112, paragraph 1. To satisfy the written description requirement, a patent disclosure must describe the claimed invention in sufficient detail that one of ordinary skill in the art can reasonably conclude that the inventor had possession of the claimed invention. MPEP § 2163(I). As discussed previously, both the '840 and the '599 disclosure describe specific embodiments of a "single instruction" for performing a "group" multiply-add operation using floating-point operands. For example, the "GF.MULADD.32" instruction is described in full detail, down to the bit level. This is a true group multiply-add instruction that has been unambiguously defined, as an illustrative embodiment of the invention. One of ordinary skill in the art would readily conclude, upon reviewing the '840 and '599 disclosures, that the Applicants had possession of the claimed invention.

To satisfy the enablement requirement, a patent disclosure when filed must contain sufficient information regarding the subject matter of the claims as to enable one of ordinary skill in the pertinent art to make and use the claimed invention. MPEP § 2164.01. Whether the enablement requirement is met depends on whether undue experimentation is necessary for one of skill in the art to practice the invention in light of the disclosure. *Id.* As discussed previously, the '840 and '599 disclosures describe specific embodiments of the claimed "group" multiply-add operation using floating-point operands, including the "GF.MULADD.32" instruction. The '840 and '599 disclosures define this instruction in a manner that specifies the precise operation of the instruction, including exactly how every bit of data is obtained (e.g., from specific registers), operated on, and presented as output. One of ordinary skill in the art

would not be required to perform undue experimentation – or any experimentation at all for that matter – to understand exactly how the "GF.MULADD.32" instruction is carried out as an illustrative embodiment of the invention. As such, the '840 and '599 disclosures clearly enable one of ordinary skill in the art to make and use the claimed invention.

Further evidence that the '840 and '599 disclosures satisfy both the written description and the enablement requirements is provided by way of a declaration from Mr. Korbin Van Dyke that is included with this Response. For example, Mr. Van Dyke states in his declaration at p. 13, paragraph 38 and pp. 14-15, paragraph 39:

**Based on the above, I believe that a person of ordinary skill in the art would readily reasonably conclude that the disclosures of the '599 patent and the '840 patent each describe the claimed [ ] elements, as recited in Claim 1 of the 10/757,851 patent application, in sufficient detail that the inventors had possession of the claimed invention, as of the August 24, 1999 filing date of the '599 patent and further as of the August 16, 1995 filing date of the '840 patent, and that the disclosures of the '599 patent and the '840 patent each provide sufficient detail to enable a person of ordinary skill in the art to make and use the claimed invention of the 10/757,851 patent application without undue experimentation as of the August 24, 1999 filing date of the '599 patent and further as of the August 16, 1995 filing date of the '840 patent.**

In fact, Mr. Van Dyke further states at p. 16, paragraph 43:

**During my evaluation of the media processor patent application, I have been impressed by the thoroughness and overall high-quality of the Zeus and Terpsichore manuals. The manuals provide clear and unambiguous descriptions of media processing systems and are thorough and well-written. The manuals provide comprehensive descriptions of instructions in complete architectural detail. The information in the manuals would have been readily understood and easily accessible to software engineers coding the media processing systems, and hardware engineers implementing microprocessors for use in the media processing systems...**

Given all the reasons stated above, Applicants respectfully submit that the pending § 103 rejections based on various combinations of Fischer, Kabir, Cohen, and Thekkath should be withdrawn. The present invention is entitled to the filing date of 8/16/95. Both the '840 and '599 disclosures provide proper support for the present invention to establish this filing date. Mr. Van Dyke's declaration is submitted in further evidence of the fact that the '840 and



Appl. No. 10/757,851  
Amdt. dated August 21, 2007  
Reply to Office Action mailed February 21, 2007

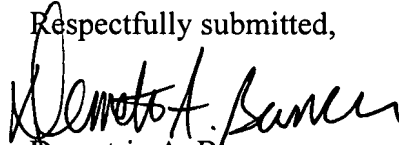
PATENT

'599 disclosures describe the invention in sufficient detail to satisfy both the written description requirement and the enablement requirement under 35 U.S.C. § 112, paragraph 1. Thus, the 8/16/95 filing date of the present invention is clearly established. Consequently, none of the cited references qualify as prior art, and the pending rejection based on these references should be withdrawn.

### CONCLUSION

In view of the foregoing, Applicants believe all claims now pending in this Application are in condition for allowance. The issuance of a formal Notice of Allowance at an early date is respectfully requested. If the Examiner believes a telephone conference would expedite prosecution of this application, please telephone the undersigned at 202-756-8335.

Respectfully submitted,



Demetria A. Buncum  
Reg. No. 58,848

600 13<sup>th</sup> Street, N.W.  
Washington, DC 20005-3096  
Phone: 202.756.8000 DAB:jrr  
Facsimile: 202.756.8087  
**Date: August 21, 2007**

**Please recognize our Customer No. 20277  
as our correspondence address.**